

Optimal Coalition Structures in Graph Games

Yoram Bachrach¹, Pushmeet Kohli¹, Vladimir Kolmogorov², Morteza Zadimoghaddam³

¹ Microsoft Research, Cambridge, UK
{yobach,pushmeet}@microsoft.com

² Institute of Science and Technology (IST), Austria vnk@ist.ac.at

³ MIT morteza@csail.mit.edu

Abstract. We consider the problem of finding the optimal coalition structure in *Weighted Graph Games (WGGs)*, a simple restricted representation of coalitional games [12]. The agents in such games are vertices of the graph, and the value of a coalition is the sum of the weights of the edges present between coalition members. The *optimal coalition structure* is a partition of the agents to coalitions, that maximizes the sum of utilities obtained by the coalitions. We show that finding the optimal coalition structure is not only hard for general graphs, but is also intractable for restricted families such as planar graphs which are amenable for many other combinatorial problems. We then provide algorithms with constant factor approximations for planar, minor-free and bounded degree graphs.

1 Introduction

Consider a set of agents who can work in teams. Some agents work well together, while others find it hard to do so. When two agents work well together, a team which contains both of them can achieve better results due to the synergy between them. However, when agents find it hard to work together, a team that contains both agents has a reduced utility due to their inability to cooperate, and may perform better when one of them is removed. How can such synergies be modeled? What teams should be formed?

Cooperation is a central issue in multi-agent systems. It is possible to analyze negotiations, contracts and strategies in such domains using *cooperative (or coalitional) games*. In such games, agents form coalitions to pursue a joint cause, and must decide how to split the gains they derive as a group. Much of previous literature explores settings where only one coalition can be formed. However, in many scenarios agents can form *multiple* disjoint coalitions, where each coalition can obtain its profits independently. An important goal is to partition the agents in a way that maximizes the total value gained by all coalitions, i.e. the social welfare. This problem is known as *finding the optimal coalitional structure*.

Representation languages for cooperative games define the value generated by each coalition. Generally any coalition may have a different value, so a naïve representation requires storage exponential in the number of agents. This emphasizes the need for a *succinct representation*. Many such representations were suggested. Some guarantee a polynomial description, but only represent restricted games (e.g. [20,12]). Others can represent any game, but require exponential storage in the worst case [17,4]. We examine coalition structures in the prominent *Weighted Graph Games (WGG)* model of [12]. In WGGs, agent synergies are expressed using a graph, where agents are vertices and the weight of an edge connecting two agents expresses how well they work together. A positive weight indicates they can coordinate well, yielding a positive contribution to a coalition containing both. The edge’s weight expresses how much utility can be derived from the cooperation of the two agents. A negative weight indicates the agents do not work well together, diminishing the utility of a team containing both. Again, the edge’s weight expresses the reduction in utility of a coalition that contains both agents.

Our contribution: We study optimal coalition structures in WGGs. We prove that finding the optimal coalition structure in WGGs is hard even for restricted families of graphs such as planar graphs. We provide constant factor approximation algorithms for planar, minor-free and bounded degree graphs.

It should be noted that the objective function that we study coincides with the well-known *Correlation Clustering* functional [6] **up to an additive constant**. Due to this additive shift existing approximability results for Correlation Clustering such as those in [26,18] do not translate to our problem. We believe that the additive normalization that we use is quite natural in our context. To our knowledge, we are the first to study approximation schemes for this functional. Furthermore, we show NP-hardness of the Correlation Clustering problem for planar graphs, which is a new result.

2 Preliminaries

A transferable utility (TU) coalitional game is composed of a set of n agents, I , and a characteristic function mapping any subset (coalition) of the agents to a rational value $v : 2^I \rightarrow \mathbb{Q}$, indicating the total utility these agents achieve together. We follow the *Coalition Structure (CS)* model of [1], where agents can form several teams working *simultaneously*. A coalition structure is a partition of the agents into disjoint coalitions. Formally, $CS = (C^1, \dots, C^l)$ is a *coalition structure* over I if $\cup_{i=1}^l C^i = I$ and $C^i \cap C^j = \emptyset$ for all $i \neq j$; The set $\mathcal{CS}(I)$ denotes all possible coalition structures on I . We overload notation and denote $v(CS) = \sum_{C^j \in CS} v(C^j)$. Given a game $\langle I, v \rangle$, an *optimal* coalition structure $CS^* \in \mathcal{CS}(I)$ is a partition that maximizes social welfare, i.e. for any other $CS \in \mathcal{CS}(I)$ we have $v(CS) \leq v(CS^*)$. We denote the problem of finding an optimal coalition structure as *OPT-CS*. OPT-CS is equivalent to a complete set partitioning problem where all disjoint subsets of the set of all agents are possible. The set partitioning problem was studied in [30], where it is shown to be NP-hard. However, we focus on OPT-CS where the inputs are restricted to WGGs [12].

Graph Games: We review the *Weighted Graph Games (WGGs)* model [12].

Definition 1. *WGGs are games played over a graph $G = \langle V, E \rangle$, with weights on the edges $w : E \rightarrow \mathbb{Q}$. The agents are the vertices, so $I = V$, and the characteristic function is the sum of the weights on the graph induced by the coalition. Given a coalition $C \subseteq V$, we denote the edges induced by the coalition as $E_C = \{e = (u, v) \in E \mid u, v \in C\}$. The characteristic function is $v(C) = \sum_{e \in E_C} w(e)$.*

As noted in [12] WGG cannot represent all games. We also assume there is at most one edge between any two vertices (parallel edges may be merged, summing the weights).

3 Finding the Optimal Coalitional Structure

We first formally define the OPT-CS problem.

Definition 2 (OPT-CS-WGG). *Given a WGG $\langle I, v \rangle$ and a rational number r , test if the value of the optimal coalitional structure for this game is at least r , i.e. if there is $CS \in \mathcal{CS}(I)$ such that $v(CS) \geq r$. We denote an optimal structure as $CS^* \in \arg \max_{CS \in \mathcal{CS}(I)} v(CS)$.*

3.1 Hardness Results

We first show that computing an optimal coalition structure in general WGGs is hard, reducing from Independent Set [15].

Definition 3 (Independent Set (IS)). *Given a graph $G = \langle V, E \rangle$, test if there is a subset of vertices of size k with no edges between them.*

Theorem 1. *OPT-CS-WGG is \mathcal{NP} -complete, and assuming $P \neq NP$, there exists no polynomial time $O(n^{1/2-\epsilon})$ -approximation algorithm for it where n is the number of vertices of the graph, and ϵ can be any positive constant. There exists no polynomial time $O(n^{1-\epsilon})$ -approximation algorithm for this problem unless $NP = ZPP$.*

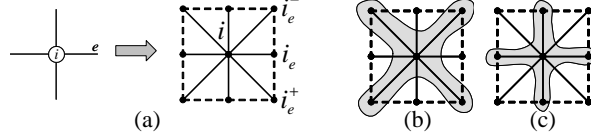


Fig. 1. (a) node $i \in V$ of degree $d = 4$ transformed to gadget G_i . (b),(c) are feasible partitionings of G_i . (b) encodes the event that i is in the independent set, while (c) encodes the opposite event.

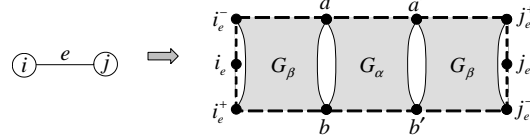


Fig. 2. Each edge $(i, j) \in E$ is transformed to gadget G_e consisting of three subgraphs G_β , G_α , G_β of figure 3.

Proof. (sketch - see appendix for full proof).

We reduce an IS instance $\langle G = \langle V, E \rangle, k \rangle$ to an OPT-CS-WGG instance. The reduced graph $G' = \langle V', E' \rangle$ contains all vertices and edges of G (so $V \subseteq V'$ and $E \subseteq E'$), and a vertex s . The weights of the original edges are all set to be $-k$ (where $k > |V|$). Vertex s is connected to each vertex in $v \in V$ with an edge e of weight $w(e) = 1$. G has an independent set of size k iff there exists a partition for graph G' with value k . This is a parsimonious reduction between IS and OPT-CS-WGG. Håstad proved that there exists no polynomial time $O(n^{1/2-\epsilon})$ -approximation algorithm for IS assuming $P \neq NP$, and that there is no polynomial time $O(n^{1-\epsilon})$ -approximation algorithm for IS unless $NP = ZPP$ [16].

The above result shows that OPT-CS-WGG has no reasonable approximation for general graphs. We show OPT-CS-WGG is hard and inapproximable even when all weights have the same absolute value, using an involved reduction from IS to the problem of OPT-CS-WGG where all edges are either $+1$ or -1 , denoted OPT-CS-WGG ± 1 .

Theorem 2. *OPT-CS-WGG ± 1 is \mathcal{NP} -complete. Assuming $P \neq NP$, there is no polynomial time $O(n^{1/2-\epsilon})$ -approximation algorithm for it where n is the number of vertices of the graph, and ϵ is any positive constant. There is no polynomial time $O(n^{1-\epsilon})$ -approximation algorithm for this problem unless $NP = ZPP$. See proof in Appendix.*

OPT-CS-WGG remains hard even for planar graphs.

Theorem 3. *OPT-CS-WGG in planar graphs is \mathcal{NP} -complete.*

Proof. We reduce from IS restricted to planar graphs, which is also \mathcal{NP} -complete [14]. Given a planar graph $G = \langle V, E \rangle$, we transform the corresponding IS instance to a planar OPT-CS-WGG instance $\langle V', E', w \rangle$. Each node $i \in V$ of degree d is replaced with gadget G_i with $2d + 1$ nodes and $4d$ edges. This gadget can be thought of as a d -sided polygon whose sides correspond to edges in $\mathcal{N}(i) \equiv \{(i, j) \in E\}$ (figure 1(a) shows an example for $d = 4$). The side corresponding to e has nodes $i_e^-, i_e, i_e^+ \in V'$ in clockwise order. If $e, e' \in E$ are consecutive edges in $\mathcal{N}(i)$ taken in clockwise order, then $i_e^+ = i_{e'}^-$. Gadget G_i also has the center node $i \in V'$ connected to all other $2d$ nodes of G_i . The edge weights are: $w(i_e^-, i_e) = w(i_e, i_e^+) = -dC$ where $C > |V|$ is a large constant, and the weights of edges from i to other nodes in G_i are $+C$, except for one edge (i, i_e^+) which has weight $C + 1$. Here edge $e \in \mathcal{N}(i)$ can be chosen arbitrarily. Each edge $e = (i, j) \in E$ is transformed to gadget G_e as shown in figure 2. G_e involves nodes $i_e^-, i_e^+, j_e^-, j_e^+$ and some internal nodes connected only to nodes in G_e .

Given a coalition structure CS in $\langle V', E' \rangle$, we denote $v_i(CS)$ and $v_e(CS)$ to be the contribution to the total cost of subgraphs G_i and G_e respectively; thus, $v(CS) = \sum_{i \in V} v_i(CS) +$

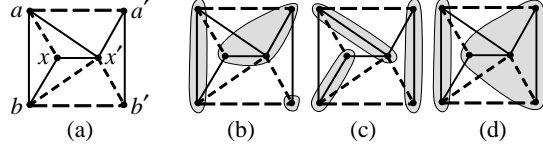


Fig. 3. (a) Topology of graphs G_α and G_β . The weights in G_α are as follows: for solid edges $w(a, b) = w(b, x) = w(x, x') = w(x', a) = w(x', a') = w(a', b') = +2C$, for internal dashed edges $w(a, x) = w(b, x') = w(x', b') = -2C$, and for horizontal side edges $w(a, a') = w(b, b') = -8C$. The weights in G_β are the same except for two edges: $w(a, b) = w(a', b') = +C$. (b), (c), (d) are optimal partitionings of G_α , and (b), (c) are optimal partitionings of G_β .

$\sum_{e \in E} v_e(CS)$. We also denote $v_i^* = \max v_i(CS)$ and $v_e^* = \max v_e(CS)$. We say that CS is *feasible for G_i* if $v_i(CS) > v_i^* - C$, and it is *feasible for G_e* if $v_e(CS) > v_e^* - C$. We denote \sim to be the equivalence relation on V' induced by CS .

Consider node $i \in V$ of degree d . It can be seen that $v_i^* = dC + 1$, and CS is feasible for G_i iff exactly one of the following holds: (1) $i_e^+ \sim i$ and $i_e \approx i$ for all $e \in \mathcal{N}(i)$ (in which case $v_i(CS) = v_i^*$), or (2) $i_e^+ \approx i$ and $i_e \sim i$ for all $e \in \mathcal{N}(i)$ (in which case $v_i(CS) = v_i^* - 1$). These cases are illustrated in figure 1(b,c). Case (1) encodes the event that i is in the independent set, and case (2) encodes the opposite event. Let S^* be a maximum independent set in $\langle V, E \rangle$. We define coalition structure CS^* as follows: (i) for each $i \in V$ select a feasible partitioning of G_i according to S ; nodes of G_i not connected to the center node i are assigned to singleton partitions. Partitions in graphs G_i, G_j for $i \neq j$ do not overlap. (ii) For each $e = (i, j) \in E$ select partitioning of G_e that does not affect the equivalence relations between nodes in $\{i_e^-, i_e^+, j_e^-, j_e^+\}$ and has the maximum possible value of $v_e(CS^*)$ among such partitionings.

Consider edge $e = (i, j) \in E$. We claim the following:

- (a) $v_e(CS^*) = v_e^*$, and thus $v(CS^*) = \sum_{i \in V} v_i^* + \sum_{e \in E} v_e^* + |S^*| - |V|$.
- (b) If CS is feasible for G_i, G_j and G_e then i and j cannot be both in the independent set, i.e. either $i \sim i_e$ or $j \sim j_e$.

These claims are proved in the Appendix. They imply that CS^* is an optimal coalition structure, so solving constructed OPT-CS-WGG instance recovers $|S^*|$. Indeed, if $v(CS) \geq v(CS^*)$ for some CS then $v(CS) > \sum_{i \in V} v_i^* + \sum_{e \in E} v_e^* - C$. Thus, $v_i(CS) > v_i^* - C$ for all $i \in V$ and $v_e(CS) > v_e^* - C$ for all $e \in V$, and so CS is feasible for all gadgets G_i and G_e . This means that CS defines some set $S \subseteq V$, and by property (b) this set is independent, implying $v(CS) \leq \sum_{i \in V} v_i^* + \sum_{e \in E} v_e^* + |S| - |V| \leq v(CS^*)$. Thus, OPT-CS-WGG in planar graphs is \mathcal{NP} -complete.

3.2 Exact Algorithms for Bounded Treewidth Graphs

For completeness, we first consider OPT-CS-WGG restricted to graphs of bounded treewidth. Lemma 1 is a special case of Lemma 2 from [8].⁴

Lemma 1. *OPT-CS-WGG where inputs are restricted to trees (or forests) is in \mathcal{P} .*

Proof. Let $P \subseteq E$ be the set of edges with positive weights, and $N \subseteq E$ be the edges with negative weights. For an edge subset A we denote its total weight as $w(A) = \sum_{e \in A} w(e)$. Note that for any structure CS we have $v(CS) \leq w(P)$. We show that for trees the optimal structure CS^* has $v(CS^*) = w(P)$. A very simple partitioning to two sub coalitions X, Y suffices for

⁴ We decided to include Lemma 1 since (i) its proof is much simpler than that of Lemma 2; (ii) it provides a distributed (local) algorithm for partitioning. Two neighbors are in the same coalition if and only if their connecting edge has positive weight; (iii) it shows that for forests, there is a simple solution that achieves all positive edges, and avoids all negative edges.

this. We begin with an arbitrary leaf v , and put it in one of the sub coalitions, so $v \in A$. For any neighbor u of v , so $e = (v, u) \in E$, we choose a sub coalition for u based on $w(e)$. If $w(e) > 0$ we put u in the same sub coalition as v , and if $w(e) < 0$ we put u in the other sub coalition. We then continue the process from the vertex u , until we deplete all the vertices in the graph. Since the graph is a tree, any edge with negative weight has one vertex in X and the other in Y , and is not counted for the value of the coalition structure, while every positive weight edge has both its vertices in the same sub coalition, and is counted for the value of the coalition structure. Thus we have $v(CS^*) = w(P)$. The partitioning can be done using a simple breadth first search, in polynomial time of $O(|V| + |E|)$.

Lemma 2 ([8]). *For k -bounded treewidth graphs (constant k), OPT-CS-WGG is in \mathcal{P} .*

3.3 Constant Factor Approximation for Planar Graphs

We now provide a polynomial time constant factor approximation algorithm for games over planar graphs. Planar graphs model many real-world situations. For example, consider coalitions between countries. In many domains, synergies would only exist between neighbouring countries. We can define a graph game where countries have an edge between them only if they are neighbours, resulting in a *planar* graph. Our method achieves a coalition structure that avoids all negative edges and gains a constant portion of the weights of the positive edges. The optimal value is at most the sum of the positive weights, yielding a constant approximation.

Definition 4. *Given a planar graph G , we denote E^+ as the set of positive edges, and E^- as the set of negative edges. A subset $E' \subseteq E^+$ is a feasible set iff there is a partition P of vertices such that any edge $e \in E'$ is contained in one part of the partition, while all negative edges are cross-part edges. We say P achieves E' .*

We find partitions that achieve feasible sets E_1, E_2, \dots, E_k (each a subset of E^+), whose union $\cup_{i=1}^k E_i$ is E^+ . Each positive edge is thus achieved by at least one of these k partitions. The value of partition i is $\sum_{e \in E_i} w(e)$ as we avoid all negative edges in our partitions, making the value of a partition be the sum of the positive weights achieved by it. The union of these k feasible sets is the set of all positive edges. Thus the sum of the values of the k partitions is at least the sum of all positive weights. Picking the maximal value partition, we obtain a k -approximation algorithm. Our algorithms finds *few* such partitions that still *cover all positive edges*.

Lemma 3. *Every matching $M \subseteq E^+$ is a feasible set. In other words, there is a partition that avoids all negative edges, and achieve edges of M .*

Proof. Let e_1, e_2, \dots, e_a be the edges of a matching. We build a partition of the vertices. We have a clusters for the a edges of our matching. We put the two endpoints of edge e_i in cluster i . There are $n - 2a$ remaining vertices as well. We put them in $n - 2a$ separate single-vertex clusters, so we achieve the edges of M in this partition. We show we avoid all negative edges. Suppose not, so there is a negative edge $e'(u, v)$ such that u and v are in the same cluster. Thus u and v are the endpoints of an edge in the matching as well, so there are two edges between u and v in to contradiction to our assumption of having no parallel edges.

Our next blocks are the union of vertex-disjoint stars using at most 3 feasible sets and forests using 6 feasible sets.

Lemma 4. *We can cover a union of several vertex-disjoint stars using at most 3 feasible sets.*

Proof. Assume there are l stars with center vertices v_1, v_2, \dots, v_l . In the star i , vertex v_i has edges to vertex set S_i . The sets S_1, S_2, \dots, S_l are disjoint and they do not contain any of the center vertices. We know that graph G is planar, and therefore we can find a proper four-coloring for it. Consider vertex v_i . Non of the vertices in set S_i has the color of vertex v_i , so the vertices

of set S_i are colored using only three colors. Without loss of generality, assume they are colored with colors 1, 2, 3. Let $S_{i,1}$ be the set of vertices in S_i with color one. Similarly we define sets $S_{i,2}$ and $S_{i,3}$. We do this for all center vertices. Note that there is no edge inside set $S_{i,j}$ for $1 \leq i \leq l$, and $1 \leq j \leq 3$. But there might be edges between different sets. Now here is the first partition that gives us the first feasible set. We put v_i and all vertices of $S_{i,1}$ in one cluster, and we do this for all center vertices. So for each center vertex, we have a separate cluster. All remaining vertices of the graph go to separate single-vertex clusters. We achieve the edges between vertex v_i , and all vertices in $S_{i,1}$ for $1 \leq i \leq l$. We avoid all negative edges because there is no edge inside set $S_{i,1}$, and there is no negative edge between vertex v_i , and set $S_{i,1}$. Similarly we use two other partitions to get the edges between v_i and sets $S_{i,2}$ and $S_{i,3}$. So we can cover all these edges using three feasible sets.

Lemma 5. *The edges of a forest can be covered using 6 feasible sets.*

Proof. We show the proof for one tree. The same thing should be done for other trees. Make the tree T rooted at some arbitrary vertex v . Now every vertex in the tree has a unique path to root vertex v . We color the edges of this tree using two colors red and blue. The edges that have odd distance to root are colored red, and the edges with even distance to root are blue. So the first level of edges that are adjacent to root are colored blue, the next level edges are red, and so on. So we start from root, and alternatively color edges blue and red. The interesting property of this coloring is that the blue edges form a union of some stars, and the same is true for red edges. Because if we consider the subgraph of blue edges, we can not find a path of length four (with three edges) in it, and we also know that there is no cycle in the graph. So these blue edges can form only a couple of stars. The same proof holds for red edges. Using Lemma 4, we can cover the blue edges with three feasible sets, and the red edges with another three feasible sets. So using at most 6 feasible sets, every edge is covered in the tree.

We decompose the vertices of graph G to several “levels” to make a hierarchical view of the graph’s structure, then use our building blocks to cover all positive edges. Let G^+ be the subgraph of G with all positive edges. Clearly G^+ is also planar so there are less than $3n$ edges in this graph (a property of planar graphs), and thus there exists at least one vertex with degree less than 6 in G^+ . We place all vertices with degree at most 5 in level 1 named L_1 , then remove them from G^+ . The remaining graph is also planar, so again there are a couple of vertices with degree less than 6. We place them in the second level, L_2 , and remove them. We repeat this process until all vertices are deleted. Suppose this process results in k levels. This implies that we have k sets L_1, L_2, \dots, L_k . There are two types of edges in G^+ : 1) edges occurring between two vertices in the *same* level, 2) edge occurring between vertices of *different* levels. First we cover the first type edges using some feasible sets. Note that for every vertex v in level i , the number of edges that connects v to vertices in level i , and upper levels $(i+1, i+2, \dots, k)$ is at most 5. Let H_1 be the subgraph of G^+ that has only edges inside levels. The degree of each vertex in H_1 is at most 5. Using Vizing Theorem [28], we can decompose the edges of H_1 into $5 + 1 = 6$ matchings in polynomial time. We can cover each of these matchings using a feasible set according to Lemma 3. Thus we take care of the inter-level edges using 6 feasible sets.

Let H_2 be the subgraph of G^+ with the remaining edges, i.e. the edges between different levels. We make all edges of H_2 directed: every edge goes from the endpoint in the lower level to the endpoint in the upper level. Thus vertices in the first level L_1 have no incoming edge, and the vertices of the top level L_k have no outgoing edge. This graph has no directed cycles. As we are moving on an edge, we are always increasing our level number. Also, the out degree of each vertex is at most 5, since the number of edges between a vertex and the upper level vertices is at most 5.

We show we can cover this graph with 5 forests. For each vertex v , we pick one of its outgoing edges arbitrarily. The selected edges form a subgraph of H_2 denoted H' . We show that H' is a forest. H' is a directed graph. The outgoing degree of each vertex in H' is at most 1 (or zero for top level vertices). It has no directed cycle and is a collection of directed rooted trees (H' is a forest). Using Lemma 5 we can cover the edges of H' by 6 feasible sets. We remove edges of

H' from H_2 . This decreases the out degree of each vertex by one (if the out degree of a vertex is zero, it remains zero). Repeating this process five times transforms the out degree of all vertices to zero, so all edges are covered. Each pass requires 6 feasible sets. Thus the total number of feasible sets we use for H_2 is at most $5 \cdot 6 = 30$. We also used 6 feasible sets for H_1 , so the total number of feasible sets is at most 36. We covered all positive edges with at most 36 feasible sets. Picking the feasible set of maximum value yields a 36-approximation.

3.4 Minor Free Graphs

We generalize our results to Minor Free Graphs. A graph H is a minor of G if H is isomorphic to a graph that can be obtained by zero or more edge contractions on a subgraph of G . G is H -Minor Free, if it does not contain H as a minor. A graph is planar iff it has no K_5 or $K_{3,3}$ as a minor [29] (K_5 is a complete graph with 5 vertices, and $K_{3,3}$ is a complete bipartite graph with 3 vertices in each part). We give an $O(h^2 \log h)$ -approximation algorithm for OPT-CS-WGG in H -minor free graphs where h is the number of vertices of graph H . This yields a constant factor approximation for planar graphs in particular, because they are K_5 -minor free graphs. We use the following theorem [27].

Theorem 4. *The number of edges of a H -Minor Free graph G with n vertices is not more than cn where c is equal to $(\alpha + o(1))h\sqrt{\log h}$, h is the number of vertices in H , and α is a constant around 0.319.*

Our algorithm for planar graphs used *sparsity* to make sure that there are low degree vertices at each level and we also need to make sure that the graph is colorable with a few colors. Using sparsity we can find a proper coloring of these graphs using $2c + 1$ colors. Any subgraph G' of G is also H -Minor Free, so its number of edges is at most $O(c|G'|)$. Thus the average degree of every subgraph of G is at most $2c$, so in every subgraph of G , we can find some vertices with degree at most $2c$ (the average degree). We use this to get a proper $2c + 1$ -coloring of G . Let v be a vertex in G with degree at most $2c$. Clearly $G \setminus \{v\}$ is also H -minor free, and inductively we can find a proper $2c + 1$ -coloring for it. Vertex v has at most $2c$ neighbors, so one of the $2c + 1$ colors is not used by its neighbors. Thus we can find one appropriate color for v among our $2c + 1$ colors, and consequently have a proper $2c + 1$ -coloring for G .

Similarly to the algorithm for planar graphs, we decompose the vertices of graph G into some levels. Every subgraph of G has some vertices with degree at most $2c$, so we take the graph G and put all vertices with degree at most $2c$ in level 1, and remove them from the graph. We then repeat this process iteratively until the graph is empty. We now consider only the edges inside levels (clusters), so the degree of each vertex is at most $2c$. We cover all these edges using at most $2c + 1$ matchings, so $2c + 1$ feasible sets are enough to take care of all edges not crossing levels (intra-level edges).

For cross-level edges, we direct the remaining graph and cover it using several forests. Again, we make every edge directed from the lower level to the upper one. Thus we make sure that the out degree of each vertex is at most $2c$, and that there is no directed cycle in the remaining graph. We can thus cover these edges using $2c$ forests, and each forest can be decomposed into two unions of stars. The entire graph G is colorable using $2c + 1$ colors, so we need $2c + 1 - 1 = 2c$ feasible sets to cover a union of stars. We need $2c \cdot 2c = 4c^2$ feasible sets to take care of the edges between levels. We conclude that $4c^2 + 2c + 1$ feasible sets are enough to cover all positive edges, and get a $4c^2 + 2c + 1 = O(c^2)$ approximation algorithm by picking the best (maximum weight) feasible set. Since c is $O(h\sqrt{\log h})$, our approximation factor is $O(h^2 \log h)$.

3.5 Bounded Degree Graphs

We now consider bounded degree graphs. A vertex's positive degree is the number of positive edges incident to it. Denote the maximum positive degree in G as Δ . Using the Vizing Theorem, the positive edges can be decomposed into $\Delta + 1$ matchings (which are also feasible sets).

This yields a $\Delta + 1$ approximation algorithm. A polynomial time algorithm for finding the decomposition can be derived from the Vizing Theorem's proof. We give a *linear time* algorithm for this problem: a random $(2 + \epsilon)\Delta$ approximation with an expected running time $O(E \log \Delta/\epsilon)$ where E is the number of edges.

We pick an arbitrary ordering of the positive edges of the graph, and try to decompose them into $(2 + \epsilon)\Delta$ matchings. We color the edges with $(2 + \epsilon)\Delta$ colors such that no two edges with the same color share an endpoint. Assume that we are in step i , and wish to color the i -th edge in our ordering. Let e be this edge with endpoints u and v . We have already colored $i - 1$ edges and some of those colored edges may have u or v as their endpoints, so we must avoid the color of those edges. For each vertex, we keep the color of its edges in a data structure. Initially the data structures for all vertices are empty. When we color an edge, we add its color to the data structure of its two endpoint vertices. We can use binary search trees to insert and search in $O(\log \Delta)$ time, since we insert at most Δ colors in each data structure. For edge e , we must find a color that is not in the union of the data structures of vertices u and v . There are at most $2(\Delta - 1)$ colors in these two data structures, and there are $(2 + \epsilon)\Delta$ colors in total. Thus if we randomly pick a color, with probability at least $\epsilon/2$, we can use this color for edge e . Checking whether a color is in a data structure can be done using a search query. Thus we can check in time $O(\log \Delta)$ whether the randomly chosen color is good or not. If the color is already taken, we can try again. It takes at most $2/\epsilon$ times in expectation to find an available color. Thus for each edge we spend $O(\log \Delta/\epsilon)$ time to find a color, so the average running time of this decomposition is $O(E \log \Delta/\epsilon)$ in expectation. Finding the best feasible set (matching) does not take more than $O(E)$ time. Thus we get an $O((2 + \epsilon)\Delta)$ approximation with almost linear running time.

4 Treewidth Based Approximations

Many hard problems are tractable for graphs with constant or bounded treewidth. We present a polynomial time $O(k^2)$ -approximation algorithm where k is the treewidth of the graph, without assuming that the treewidth of graph G is constant or a small number. Algorithms for finding the treewidth of a graph only work in polynomial time when the treewidth is constant. Although we do not know the treewidth, we can still make sure that the approximation factor is not more than $O(k^2)$. We use the following lemma.

Lemma 6. *If G has treewidth k , it has at least one vertex with degree at most k .*

Proof. Since the treewidth of G is k , there is a tree T such that every vertex of T has a subset of size at most $k + 1$ of vertices of G . We know that the vertices of T that contain a vertex of G form a connected subtree. We also know that the endpoint vertices of each edge in G are in the set of at least one vertex of T together. Now we can prove our claim. Consider a leaf of T live v . Let u be the father of v . These two vertices have two subsets of vertices of G like S_u and S_v . If S_v is a subset of S_u , there is no need to keep vertex v in our tree. We can delete it from T , and the remaining tree is also a proper representation of graph G . So we know that there is at least a vertex of G like x which is in S_v , and not in S_u . Clearly v is the only vertex of T that contains x . Otherwise the vertices of T that contain x do not form a connected subgraph. So vertex x can have neighbors only in set S_v which means that x has at most $k + 1 - 1 = k$ neighbors.

Removing a vertex from a graph does not increase its treewidth, so we can iteratively find vertices of degree at most k , and delete them. This way we can find a $(k + 1)$ proper coloring of vertices of G . We use the same decomposition we used for planar graphs, so we decompose the positive edges into $O(k)$ matchings and unions of stars. Each matching is a feasible set, and each union of stars can be decomposed into $k + 1 - 1 = k$ feasible sets, as we can color the graph with $k + 1$ colors. Thus $O(k^2)$ feasible sets are enough to cover all positive edges yielding an $O(k^2)$ approximation algorithm. Note that we start with a value of k , and we keep deleting vertices with degree at most k . If every vertex is deleted after some number of iterations, we achieve the desired structure. Otherwise at some point the degree of each vertex is greater than

k , indicating that the treewidth is more than k . Thus, we can find the minimum k for which every vertex will be deleted after some steps, and that k is at most the treewidth of G .

5 Related Work

Cooperation is central in designing multi-agent systems. Several papers describe representations of cooperative domains based on combinatorial structures [12,17,4] and a survey in [7]. Generation of the optimal coalition structure received much attention [22,21] along with its applications, such as vehicle routing [23] and multi-sensor networks [9]. An early approach by [25], focused on overlapping coalitions and gave a loose approximation algorithm. Another early approach [24] has a worst case complexity of $O(n^n)$, whereas dynamic programming approaches [30] have a worst case guarantee of $O(3^n)$. Arguably, the state of the art method is presented in [21]. They suggest an algorithm with a worst case runtime of $O(n^n)$ and do not provide polynomial guarantees even for restricted classes, but in practice it is faster than the above mentioned methods. These approaches assume a black-box that computes the value of a coalition, whereas we rely on a specific representation. Another approach solves the coalition structure generation problem [3], but relies on a different representation. Another approach has been suggested for typed-games [2], where agents fall into a bounded number of equivalence classes (the running time is exponential in the number of such types). However, in the graph games we study the number of agent types is unbounded, so this approach is untractable. The approach of [9], which suggests a polynomial algorithm for a restricted case of multi-sensor networks, is also somewhat similar to our method, but uses a different representation. Other work of similar nature that uses different representation languages includes [13,19]. In contrast to the above approaches, we provide polynomial algorithms and sufficient conditions that guarantee various *approximation ratios* for the prominent WGG representation [12].

6 Conclusion

We considered computing the optimal coalitional structure in WGGs. We showed that although the problem is \mathcal{NP} -hard, further restrictions on the input graph, such as being a tree or having bounded treewidth result in tractable algorithms for the problem. We also showed the problem is hard for planar graphs, but provided a polynomial constant approximation algorithm for this class and other classes. WGGs are a well known representation of cooperative games, and offer a simple and concise way of expressing synergies. However, the WGG representation cannot express all cooperative game, so our methods are not suitable to all such game. We believe, however, that many interesting domains can be expressed as WGGs.

Several directions remain open for future research. First, since solving the coalition structure generation problem is hard for general WGGs, are there alternative approximation algorithms for doing so? Obviously, one can use the algorithms for general games, however we believe a much better complexity can be achieved for WGGs than for general games. Second, focusing on the game theoretic motivation for this work, it would be interesting to examine *core-stable* coalition structures rather than just optimal ones. It is not known whether there exists a PTAS⁵ for planar graphs or not. Though one might hope to obtain such a PTAS by combining Baker's approach with our algorithm for solving bounded treewidth graphs, such a direct approach fails⁶. Finally, it would be interesting to examine other classes of graphs where it is possible to solve the coalition structure generation in polynomial time. Also, similar tractability results for coalition structure generation could be devised by using other representation languages.

References

1. RJ Aumann and JH Dreze. Cooperative games with coalition structures. *International Journal of Game Theory*, 3(4):217–237, 1974.

⁵ Polynomial-Time Approximation Scheme.

⁶ See appendix for a complete discussion.

2. H. Aziz and B. de Keijzer. Complexity of coalition structure generation. *Arxiv preprint arXiv:1101.1007*, 2011.
3. Y. Bachrach, R. Meir, K. Jung, and P. Kohli. Coalitional Structure Generation in Skill Games. In *AAAI-2010*, 2010.
4. Yoram Bachrach and Jeffrey S. Rosenschein. Coalitional skill games. In *AAMAS 2008*, pages 1023–1030, Estoril, Portugal, May 2008.
5. Brenda S. Baker. Approximation algorithms for np-complete problems on planar graphs. *J. ACM*, 41:153–180, January 1994.
6. N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning Journal*, pages 86–113, 2004.
7. J. M. Bilbao. *Cooperative Games on Combinatorial Structures*. Kluwer Publishers, 2000.
8. P. J. Cowans and M. Szummer. A graphical model for simultaneous partitioning and labeling. In *AISTATS*, 2005.
9. V.D. Dang, R.K. Dash, A. Rogers, and N.R. Jennings. Overlapping coalition formation for efficient data fusion in multi-sensor networks. In *AAAI-2006*, volume 21, page 635, 2006.
10. E. D. Demaine, M. T. Hajiaghayi, and K. Kawarabayashi. Algorithmic graph minor theory: Decomposition, approximation, and coloring. In *FOCS*, pages 637–646, 2005.
11. Erik D. Demaine and Mohammad Taghi Hajiaghayi. Diameter and treewidth in minor-closed graph families, revisited. *Algorithmica*, 40:211–215, 2004.
12. Xiaotie Deng and Christos H. Papadimitriou. On the complexity of cooperative solution concepts. *Math. Oper. Res.*, 19(2):257–266, 1994.
13. E. Elkind, G. Chalkiadakis, and N.R. Jennings. Coalition structures in weighted voting games. In *ECAI-2008*, page 393, 2008.
14. M. Garey, D. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoret. Comput. Sci.*, 1:237–267, 1976.
15. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.
16. J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *FOCS*, pages 627–636, 1996.
17. Samuel Yeong and Yoav Shoham. Multi-attribute coalitional games. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 170–179, 2006.
18. P. Mitra and M. Samal. Approximation algorithm for correlation clustering. In *Networked Digital Technologies (NDT)*, 2009.
19. N. Ohta, V. Conitzer, R. Ichimura, Y. Sakurai, A. Iwasaki, and M. Yokoo. Coalition Structure Generation Utilizing Compact Characteristic Function Representations. In *CP-09*, 2009.
20. B. Peleg and P. Sudholter. *Introduction to the theory of cooperative games*. Springer, 2007.
21. T. Rahwan and N.R. Jennings. An improved dynamic programming algorithm for coalition structure generation. In *AAMAS*, pages 1417–1420, 2008.
22. T. Rahwan, S.D. Ramchurn, V.D. Dang, A. Giovannucci, and N.R. Jennings. Anytime optimal coalition structure generation. In *AAAI*, volume 22, page 1184, 2007.
23. T.W. Sandholm and V.R.T. Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1-2):99–137, 1997.
24. T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1):209–238, 1999.
25. Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, 1998.
26. Jinsong Tan. A note on the inapproximability of correlation clustering. *Information Processing Letters*, 108(5):331–335, 2008.
27. Andrew Thomason. The extremal function for complete minors. *J. Comb. Theory Ser. B*, 81(2):318–338, 2001.
28. V.G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz*, 3(7):25–30, 1964.
29. K. Wagner. Über eine Eigenschaft der ebenen Komplexe. *Math. Ann.*, 114:570–590, 1937.
30. D. Yun Yeh. A dynamic programming approach to the complete set partitioning problem. *BIT Numerical Mathematics*, 26(4):467–474, 1986.

7 Appendix I: Detailed Proofs

Theorem 1 OPT-CS-WGG is \mathcal{NP} -complete, and assuming $P \neq NP$, there exists no polynomial time $O(n^{1/2-\epsilon})$ -approximation algorithm for it where n is the number of vertices of the graph, and ϵ can be any positive constant. There exists no polynomial time $O(n^{1-\epsilon})$ -approximation algorithm for this problem unless $NP = ZPP$.

Proof. We reduce an Independent Set instance $G = \langle V, E \rangle$ to a OPT-CS-WGG instance. The reduced graph $G' = \langle V', E' \rangle$ contains all the vertices and edges of G (so $V \subseteq V'$ and $E \subseteq E'$), and one additional vertex s . The weights of the original edges are identical, and are all $w(e) = -k$ where $k > |V|$. Vertex s is connected to any vertex in $v \in V$ with an edge e with weight $w(e) = 1$. Graph G has an independent set of size k iff there exists a partition for graph G' with value k .

If G has an independent set S with size k , we can create a coalition structure CS with value k as follows. We put (the additional) vertex s , and the k vertices in set S in one coalition. For every other vertex $v \notin S$, we make a separate coalition with only vertex v in it, so the rest of the vertices are put in $n - k$ separate coalitions where n is the number of vertices of graph G . A coalition with one vertex has value zero. Thus the value of the whole coalition structure is equal to the value of the coalition that includes s . There is no negative edge between vertices of this coalition, because set S is an independent set in graph G . There are k edges with weight $+1$ in this coalition, so the value of the coalition structure is equal to k .

On the other hand, if the optimal coalition structure has value k' in graph G' , we can find an independent set of size k' in G . We note that the absolute value of a negative edge is more than the sum of all positive edge weights because there are $n = |V|$ positive edges with weight $+1$. We have the option of putting each vertex in a separate coalition and achieving value zero. Thus, we never put a negative edge in a coalition in the optimal coalition structure, as this obtains a negative value. Thus, the value of the optimal coalition structure is the number of positive edges we get through the coalitions. However, all positive edges are between vertex s and the other vertices, so the value of the coalitions is the number of vertices from graph G that we put in the coalition of vertex s . We can safely put the rest of the vertices in separate coalitions as there are no positive edges between them, so there is no benefit in putting them in the same coalition. In the optimal solution the vertices we put in the same sub-coalition as vertex s have no negative edges between them, so they form an independent set in graph G .

From the analysis, we conclude that the above-mentioned is a parsimony reduction between the independent set problem and the OPT-CS-WGG problem. Håstad proved that there exists no polynomial time $O(n^{1/2-\epsilon})$ -approximation algorithm for independent set problem assuming $P \neq NP$, he also proved that there is no polynomial time $O(n^{1-\epsilon})$ -approximation algorithm for it unless $NP = ZPP$ [16]. So the same hardness results work for the OPT-CS-WGG problem. For additional discussion of optimal coalition structures in graph games, see [2].

Theorem 2 OPT-CS-WGG ± 1 is \mathcal{NP} -complete, and assuming $P \neq NP$, there exists no polynomial time $O(n^{1/2-\epsilon})$ -approximation algorithm for it where n is the number of vertices of the graph, and ϵ can be any positive constant. There exists no polynomial time $O(n^{1-\epsilon})$ -approximation algorithm for this problem unless $NP = ZPP$.

Proof. Similarly to the proof of Theorem 1, we reduce an Independent Set instance $G = \langle V, E \rangle$ to a OPT-CS-WGG instance. The reduced graph $G' = \langle V', E' \rangle$ contains all the vertices and edges of G (so $V \subseteq V'$ and $E \subseteq E'$), and one additional vertex s . The weights of the original edges are identical, and are all $w(e) = -1$ (this is where the reduction differs from that of Theorem 1). Vertex s is connected to any vertex in $v \in V$ with an edge e with weight $w(e) = +1$.

If the optimal coalition structure has value k' in graph G' , we show we can find an independent set of size at least $k'/9$ in G . As in the previous reduction, all positive edges are incident to vertex s . Thus every vertex is either in the coalition of vertex s or in a separate single-vertex coalition. Let S be the set of vertices from graph G in the coalition of vertex s . The value of coalition S is thus the value of the entire coalition structure. Suppose there are a vertices in S , and b negative edges between vertices of S . In this case, the sum of all weights of positive edges for S is equal to a , and the sum of all weights of negative edges in S is $-b$. Therefore, $k' = a - b$, so $a > k'$. Also note that $b < a$.

Now consider the subgraph $G[S]$. The number of edges in this graph, b , is not more than the number of vertices a . Therefore, the average degree is at most 2 in this subgraph. Thus, the number of vertices with degree at least 3 in this subgraph is not more than $2a/3$, so there are at least $a/3$ vertices with degree at most 2. Let S' be the set of vertices with degree at most 2 in the subgraph $G[S]$. Now consider the subgraph $G[S']$. This subgraph has at least $a/3$ vertices, each with a degree at most 2. We can pick $1/3$ of

the vertices of this subgraph as an independent set: we just pick a vertex arbitrarily, put it in our independent set and remove its neighbors, which are no more than two vertices. Thus, in the worst case, for every three vertices, we choose one for our independent set. This way, we can find an independent set of size at least $a/9 \geq k'/9$, so our reduction loses a factor of at most 9, but the same hardness results hold asymptotically.

Theorem 3 OPT-CS-WGG in planar graphs is \mathcal{NP} -complete.

Proof. We use a reduction from the Independent Set problem restricted to planar graphs, which is also \mathcal{NP} -complete [14]. Given a planar graph $G = \langle V, E \rangle$, we transform the corresponding Independent Set instance it to the following planar OPT-CS-WGG instance $\langle V', E', w \rangle$. Each node $i \in V$ of degree d is replaced with gadget G_i with $2d + 1$ nodes and $4d$ edges. This gadget can be thought of as a d -sided polygon whose sides correspond to edges in $\mathcal{N}(i) \equiv \{(i, j) \in E\}$ (figure 1(a) shows an example for $d = 4$). The side corresponding to e has nodes $i_e^-, i_e, i_e^+ \in V'$ arranged in the clockwise order. Note, if $e, e' \in E$ are two consecutive edges in $\mathcal{N}(i)$ taken in the clockwise order then $i_e^+ = i_{e'}^-$. Gadget G_i also has the center node $i \in V'$ connected to all other $2d$ nodes of G_i . The edge weights are as follows: $w(i_e^-, i_e) = w(i_e, i_e^+) = -dC$ where C is a sufficiently large constant (namely, $C > |V|$), and the weights of edges from i to other nodes in G_i are $+C$, except for one edge (i, i_e^+) which has weight $C + 1$. Here edge $e \in \mathcal{N}(i)$ can be chosen arbitrarily.

Each edge $e = (i, j) \in E$ is transformed to gadget G_e as shown in figure 2. Note, G_e involves nodes $i_e^-, i_e^+, j_e^-, j_e^+$ and some internal nodes which are connected only to nodes in G_e .

Given a coalition structure CS in $\langle V', E' \rangle$, we denote $v_i(CS)$ and $v_e(CS)$ to be the contribution to the total cost of subgraphs G_i and G_e respectively; thus, $v(CS) = \sum_{i \in V} v_i(CS) + \sum_{e \in E} v_e(CS)$. We also denote $v_i^* = \max v_i(CS)$ and $v_e^* = \max v_e(CS)$. We say that coalition structure CS is *feasible for* G_i if $v_i(CS) > v_i^* - C$, and it is *feasible for* G_e if $v_e(CS) > v_e^* - C$. We denote \sim to be the equivalence relation on V' induced by CS .

Consider node $i \in V$ of degree d . It can be seen that $v_i^* = dC + 1$, and CS is feasible for G_i iff exactly one of the following holds: (1) $i_e^+ \sim i$ and $i_e \sim i$ for all $e \in \mathcal{N}(i)$ (in which case $v_i(CS) = v_i^*$), or (2) $i_e^+ \sim i$ and $i_e \sim i$ for all $e \in \mathcal{N}(i)$ (in which case $v_i(CS) = v_i^* - 1$). These cases are illustrated in figure 1(b,c). Case (1) will encode the event that i is in the independent set, and case (2) will encode the opposite event.

Let S^* be a maximum independent set in $\langle V, E \rangle$. We define coalition structure CS^* as follows: (i) for each $i \in V$ select a feasible partitioning of G_i according to S^* ; nodes of G_i not connected to the center node i are assigned to singleton partitions. Partitions in graphs G_i, G_j for $i \neq j$ do not overlap. (ii) For each $e = (i, j) \in E$ select partitioning of G_e that does not affect the equivalence relations between nodes in $\{i_e^-, i_e^+, j_e^-, j_e^+\}$ and has the maximum possible value of $v_e(CS^*)$ among such partitionings.

Consider edge $e = (i, j) \in E$. We prove below two facts:

(a) $v_e(CS^*) = v_e^*$, and thus

$$v(CS^*) = \sum_{i \in S^*} v_i^* + \sum_{i \in V - S^*} (v_i^* - 1) + \sum_{e \in E} v_e^* = \sum_{i \in V} v_i^* + \sum_{e \in E} v_e^* + |S^*| - |V|.$$

(b) If CS is feasible for G_i, G_j and G_e then i and j cannot be both in the independent set, i.e. either $i \sim i_e$ or $j \sim j_e$.

This will imply that CS^* is an optimal coalition structure, and so solving constructed OPT-CS-WGG instance will recover $|S^*|$. Indeed, if $v(CS) \geq v(CS^*)$ for some CS then $v(CS) > \sum_{i \in V} v_i^* + \sum_{e \in E} v_e^* - C$. Thus, $v_i(CS) > v_i^* - C$ for all $i \in V$ and $v_e(CS) > v_e^* - C$ for all $e \in V$, and so CS is feasible for all gadgets G_i and G_e . This means that CS defines some set $S \subseteq V$, and by property (b) this set is independent, implying $v(CS) \leq \sum_{i \in S} v_i^* + \sum_{i \in V - S} (v_i^* - 1) + \sum_{e \in E} v_e^* = \sum_{i \in V} v_i^* + \sum_{e \in E} v_e^* + |S| - |V| \leq v(CS^*)$.

To prove (a) and (b), we first analyze optimal partitionings of graphs G_α and G_β specified in figure 3.

Graph G_α Let $v_\alpha(CS)$ be the contribution of G_α to the total cost. If the restriction of CS to G_α is as shown in figure 3(b,c,d) then $v_\alpha(CS) = 6C$. This is the maximum possible cost, i.e. $v_\alpha^* \equiv \max v_\alpha(CS) = 6C$. Indeed, the subgraph induced by nodes $\{a, b, x, x'\}$ cannot contribute cost larger than $4C$, and the subgraph induced by nodes $\{a', b', x'\}$ cannot contribute cost larger than $2C$. Clearly, we must have $a \sim a'$ and $b \sim b'$, otherwise $v_\alpha(CS) \leq v_\alpha^* - 2C$. Assuming that these conditions hold, 4 cases are possible:

- (i) $a \sim b, a' \sim b'$;
- (ii) $a \sim b, a' \sim b'$;

- (iii) $a \sim b, a' \sim b'$;
- (iv) $a \approx b, a' \approx b'$.

In the first 3 cases there exist a partitioning CS of G_α with $v_\alpha(CS) = v_\alpha^*$ (see figure 3(b,c,d)). In case (iv), however, we have $v_\alpha(CS) \leq v_\alpha^* - 2C$. Indeed, edges (a, b) and (a', b') are not contributed, and subgraphs induced by $\{a, b, x, x'\}$ and $\{a', b', c'\}$ contribute not more than $4C$ and $2C$ respectively. Furthermore, if the first subgraph contributes more than $2C$ then we must have $a \sim x'$ and thus $a' \approx x'$, so the second subgraph does not contribute.

Graph G_β Again we must $a \approx a'$ and $b \approx b'$, otherwise $v_\beta(CS) \leq v_\beta^* - C$. Assume that these conditions hold. Then in cases (i), (ii) there exist a partitioning CS of G_β with $v_\beta(CS) = v_\beta^*$ (as shown in figure 3(b,c)), and in cases (iii), (iv) we have $v_\alpha(CS) \leq v_\alpha^* - C$. This can be derived by observing that G_β is obtained from G_α by subtracting weight C from edges (a, a') and (b, b') , and applying the analysis for G_α above.

We are now ready to prove properties (a) and (b). We use the notation from figure 2. It can be checked $v_e^* = v_\beta^* + v_\alpha^* + v_\beta^*$, and we have the following: CS is feasible for $G_e \Leftrightarrow v_e(CS) = v_e^* \Leftrightarrow v_\alpha(CS) = v_\alpha^*, v_\beta(CS) = v_\beta^*$ for both copies of graph G_β involved in G_e .

(a) Let \sim be the equivalence relation induced by CS^* . Three cases are possible:

- $i \notin S^*, j \notin S^*$. Then $i_e^- \approx i_e^+, j_e^- \approx j_e^+$, and so we can choose a partitioning CS of G_e with $v_e(CS) = v_e^*$ by compositing optimal partitionings (c), (d), (b) from figure 3 for the three subgraphs $G_\beta, G_\alpha, G_\beta$ respectively. Clearly, these partitionings are consistent with each other; we have $a \sim b$ and $a' \sim b'$.
- $i \in S^*, j \notin S^*$. Then $i_e^- \sim i_e^+, j_e^- \approx j_e^+$, and so we can choose a partitioning CS of G_e with $v_e(CS) = v_e^*$ by compositing optimal partitionings (b), (c), (b) from figure 3 for the three subgraphs $G_\beta, G_\alpha, G_\beta$ respectively. Clearly, these partitionings are consistent with each other; we have $a \approx b$ and $a' \sim b'$.
- $i \notin S^*, j \in S^*$. Then $i_e^- \approx i_e^+, j_e^- \sim j_e^+$, and so we can choose a partitioning CS of G_e with $v_e(CS) = v_e^*$ by compositing optimal partitionings (c), (b), (c) from figure 3 for the three subgraphs $G_\beta, G_\alpha, G_\beta$ respectively. Clearly, these partitionings are consistent with each other; we have $a \sim b$ and $a' \approx b'$.

(b) Suppose that CS is feasible for G_i, G_j, G_e and i, j are both in the independent set implying $i_e^- \sim i_e^+$ and $j_e^- \sim j_e^+$. For graphs G_β we can only have cases (i) and (ii), therefore $a \approx b$ and $a' \approx b'$. Thus, we have case (iv) for graph G_α - a contradiction.

8 Appendix II: Why Baker's Approach [5] Does Not Yield a PTAS for Planar Graphs

Baker's approach [5] and other decomposition theorems [10] have been used to partition vertices or edges of a planar graph and more generally a minor free graph into an arbitrary number of sets such that removing any of these sets results a bounded treewidth graph. Many problems are tractable for bounded treewidth graphs, so this technique is quite powerful for domains where each set in the decomposition is small enough so that deleting it does not cause extreme changes in the optimal solution. One might hope that the same approach would allow obtaining a PTAS for the optimal coalition structure in planar graphs. We show why a direct solution based on Baker's approach can not be used to get a PTAS, and discuss why other decomposition theorems are not helpful in this setting as well.

We first briefly explain Baker's approach of finding disjoint subsets of edges of the graph such that the deletion of any of these sets of edges results a bounded treewidth graph. Take an arbitrary vertex v , and consider the BFS tree rooted at v . Let L_i be the set of vertices at distance i from v , let l be the number of these sets, i.e. l is the maximum distance from v . These sets are called "layers". For any integer number k , and $i < k$, if we remove the edges between sets L_{i+jk} and L_{1+i+jk} for all $0 \leq j \leq l/k$, the resulting graph would have treewidth $O(k)$ (see [11] for a proof). Therefore removing edges between layers L_{i+jk} and L_{1+i+jk} for all $0 \leq j \leq l/k$ gives us a bounded treewidth graph for which we have a polynomial time exact algorithm.

Let E_i be the set of edges between layers L_{i+jk} and L_{1+i+jk} for all $0 \leq j \leq l/k$. Baker's approach provides k disjoint sets E_0, E_1, \dots, E_{k-1} such that deleting any E_i gives us an $O(k)$ bounded treewidth graph. If the sets generated by Baker's algorithm were a partition, we would get a PTAS for planar graphs. Unfortunately, these k sets do not form a partition of edges of the graph. They are disjoint, but their union

is *not* all the edges, since there may be edges inside each layer which are not in any of these k sets. We now discuss how such an approach would work if $\{E_i\}_{i=0}^{k-1}$ were a partition and explain where it fails when $\{E_i\}_{i=0}^{k-1}$ is not a partition.

Consider the case where the above $\{E_i\}_{i=0}^{k-1}$ is a partition. For any i , we can modify the optimal solution to avoid all edges in set E_i , while decreasing the solution's quality. Assume that the optimum solution partitions the vertices of the graph into t sets S_1, S_2, \dots, S_t . Let E_{opt} be the set of edges covered in this optimum solution. When removing the edges in E_i we obtain a graph with some connected components. In each connected component, we can use the optimum solution's partitioning to partition the vertices. We use the projection of the optimum solution's partition in each connected component. In this new partitioning, we cover all edges that the optimum solution covers except the ones in E_i . Note that this new partition is a feasible solution in the graph $G \setminus E_i$ (the graph G after removing the edges in E_i). Since graph $G \setminus E_i$ has bounded treewidth, we can find its optimum solution in polynomial time. This solution has value at least $\sum_{e \in E_{opt}} w(e) - \sum_{e \in E_{opt} \cap E_i} w(e)$ (the value of the solution derived by projecting the optimum solution in each connected component of $G \setminus E_i$). We can find the optimum solution in $G \setminus E_i$ for each $0 \leq i < k$. The average of these k solutions is at least:

$$\frac{\sum_{i=0}^{k-1} \left[\sum_{e \in E_{opt}} w(e) - \sum_{e \in E_{opt} \cap E_i} w(e) \right]}{k} = \frac{k \sum_{e \in E_{opt}} w(e) - \sum_{i=0}^{k-1} \sum_{e \in E_{opt} \cap E_i} w(e)}{k}$$

Unfortunately, the sets $\{E_i\}_{i=0}^{k-1}$ are disjoint but do not form a partition. An approach based on Baker's algorithm works when $\sum_{i=0}^{k-1} \sum_{e \in E_{opt} \cap E_i} w(e)$ is at most $\sum_{e \in E_{opt}} w(e)$. However, this may not hold since there are negative edges in the graph, and possibly in the optimum solution as well. It might be the case that the positive edges in the optimum solution are shared with sets E_i 's, and the negative edges of the optimum solution are not in any set E_i . In this case, we can not obtain any good upper bound on $\sum_{i=0}^{k-1} \sum_{e \in E_{opt} \cap E_i} w(e)$, and it may even be larger than $k \sum_{e \in E_{opt}} w(e)$, so we may gain nothing through this approach.

The key problem in the above approach is that the sets of edges in Baker's approach do not form a partition of edges of the graph: they are disjoint sets, but do not span all the edges. In particular, the edges between two consecutive layers are covered by Baker's approach, and the edges inside each layer are not present in any set. Thus, if all edges between two layers have weight $+\infty$ and the edges inside each layer have weight $-\infty$, we can not find any reasonable upper bound on $\sum_{i=0}^{k-1} \sum_{e \in E_{opt} \cap E_i} w(e)$ in terms of the optimum solution.

Other decomposition theorems, such as [10], are also not very helpful for similar reasons. For example, Theorem 3.1 in [10] states that we can partition the vertices of a minor free graph into k sets such that the deletion of any of these sets results a bounded treewidth graph. One might hope there exists one of these k sets whose deletion does not effect the optimal solution by a factor of more than $1 - 1/k$. A counterexample is when all positive edges in the optimal solution are between these k sets, and the negative edges of the optimal solution are inside these k sets. Again, the absolute value of the weights of all edges may be much larger than the value of the optimal solution. In this case, removing these k sets affects the positive edges more than the negative edges, which can have a huge effect the value of the solution. In particular, if we remove each of these k sets once, every (positive) edge between two sets is "removed" twice (for each of its endpoints), and every (negative) edge inside a set is "removed" only once.